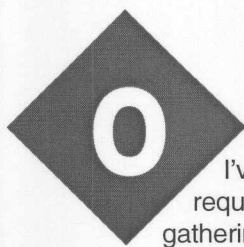


LESSONS FROM THE TRENCHES

George Martin

Off-the-Shelf Data Acquisition Using Visual Basic

Rest easy all you spotted owl fans, George isn't headed for the old growth forests; he's headed toward the new market for data-acquisition systems. As desktop spreadsheet and data-graphing applications become more user friendly, the market for custom data-logging systems is becoming tighter. Join George as he takes a data-acquisition module, Visual Basic, and whittles down some code to form a general-purpose data logger.



Over the years, I've had several requests for data-gathering programs, typically for factory-testing applications. Each assembly was tested, and results were printed in report form and filed away for trend analysis. Ten years ago, it was a difficult task, requiring lots of custom programming, which meant I had a reasonable chance of winning the contract.

Recently, however, I've observed a change in the requests. Today's typical user has a spreadsheet and can graph data quite well, thank you very much. Also, most data-acquisition systems advertise easy-to-use interfaces.

Factor in the lower cost and higher power of today's desktop computers, and you've got a changed market. Fewer companies are willing to spend large amounts of money on custom hardware and software development. You have the choice to adapt or die.

This month, I'd like to participate in this new world order or at least the embedded corner of it. I've been threatening to write a general-purpose data logger, so here we go. My goal is to take an off-the-shelf,

data-acquisition board, combine it with some Visual Basic (VB) software, and chart the data. I'd like to develop generic code for this article so that you can easily alter my work to suit your needs. (If you wish to compile the code, [download the source code](#) because it has not been altered, as the Listings have been, for viewing.)

I've done some work with the [Micromint Answer MAN](#) modules and would like to use them as the basis for the data-acquisition system. They're low cost and powerful. I'll let you decide if they're easy to use.

I'm going to use a modular design for the critical interfaces to all the major components so they can be easily rewritten. VB requires a modular discipline to develop a good design. Adapting it to your hardware should be straightforward.

PROGRAM REQUIREMENTS

Let me start by defining the requirements. I need:

1. Simple controls so no user's manual is required
2. A setup screen
3. A run screen
4. Interface to data acquisition via the serial port
5. Code specific to a segregated module
6. Data gathered and placed in a file for analysis by other programs
7. The use of VB's Microsoft Chart form to view the data as it's gathered
8. Enough flexibility so that the code can be used as is

By the way, I don't believe anyone can write a program or do any type of design work without a set of requirements. And, if you can't write your requirements down, they aren't clear enough. As engineers, you will encounter customers or marketing departments that want everything. Start with a list. If you or your customer is ISO-900x certified, then starting with a list is the law.

So far, so good. Break the design into the following components:

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;ANSWER MAN DATA LOGGER VERSION 00.04
;OPENED @ DATE: 4/18/99 TIME: 2:09:52 PM
;SAMPLEPERIOD: 0.5 (SEC)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
-1      -1      -1      -1      0      -1
-1      -1      -1      -1      0      -1
-1      -1      -1      -1      0      -1
-1      -1      -1      -1      0      -1
-1      -1      -1      -1      0      -1
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;CLOSED @ DATE: 4/18/99 TIME: 2:09:55 PM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

```

Figure 1—Catch a glimpse of the straightforward output you can get from this tadpole.

1. Setup (e.g., comm ports, data rate, and file names)
2. Real-time operation (e.g., data gathering, filing the results)
3. Charting the data

I've never used Microsoft Chart control, and it looks like a bear. Perhaps, a friendly bear, but it's obviously rich in features, choices, options, and places to make errors. I'll have to make this a two-part article. Keep in mind that, until the charting is complete, I will be changing the code published in this article. I've posted notes in the code so you won't be surprised.

GETTING IT ALL SET UP

I am using VB5 Service Pack 3 and running Windows 95. VB in Windows 95 is an interesting programming environment. The language is now object oriented, so you can use objects in your design. But, it also permits you to construct programs just like you would have done before objects.

There is, however, one change you can't ignore in Win95. In the old days (DOS), your program would have the machine all to itself. You could loop your code until a key was pressed. In fact, it made for a snappy response because the machine and the operating system weren't doing anything else anyway.

But, if a VB module loops on itself and never frees the machine for other tasks, you lose system ser-

vices such as the mouse and keyboard. Instead of looping to wait for an action, you need to assemble a collection of routines that execute whenever a specific event occurs. Events are key pressed, mouse clicked, or timer expired. The operating system informs VB of the event and places you at the start of the routine. You then run the routine and, when you exit from it, control returns to the operating system. This programming structure takes some getting used to, but my programs are improving with familiarity. The VB routines are contained in modules. A module can just be a collection of routines, or a form can be associated with the module. The visual controls and displays are placed in the form.

I've got to admit that this is a very easy way to create a form that

performs a real function. As with any powerful tool, you can get carried away. Keep it simple. Remember there is no user's manual with this project.

The structure of this project will start out containing these modules:

Modules with associated forms:

- TopForm Contains the start screen setting all the variables
- PickFile Contains the code to select a data file
- Charting Contains the code to chart the data

Modules without forms associated:

- AnswerManStruct Defines the structure interface to the Answer MAN DLL
- Globals Contains the global definitions (see Listing 1)

PROGRAMMING THE FUNDAMENTAL MODULES

Listing 2 provides, TopForm, the first module and form. In it, I specify these parameters:

- >Comm port
- >Data rate
- >Module name
- >Channels to sample
- >Update rate
- >Gather data start/stop
- >Append/overwrite
- >Exit

Each button causes the associ-

Listing 1—In Visual Basic, routines are contained in modules and may or may not have a form associated with them. The Globals routine has no form as it simply tracks the global definitions needed by the modules.

```

Attribute VB_Name = "Globals"
'
' CAUTION This module is NOT the final version.
'
Option Explicit
' Color Constants

Public Const COLOR_BLACK = &H0&Public Const COLOR_RED =
&HFF&Public Const COLOR_GREEN = &HFF00&Public Const COLOR_YELLOW
= &HFFFF&Public Const COLOR_WHITE = &HFFFFFF
Public Const COLOR_GRAY = &HCOCOCO
Public Const COLOR_ORANGE = &H80FF&

```

Listing 2—TopForm is the first module and it contains the start screen.
TopForm sets variables, such as communications port, data rate, number of channels to sample, and update rate.(continued on following page)

```

VERSION 5.00
Begin VB.Form TopForm
    Caption       = "Data Logger V00-01"
    ClientHeight  = 6024
    ClientLeft    = 1008
    ClientTop     = 1968
    ClientWidth   = 11220
    LinkTopic     = "Form1"
    ScaleHeight   = 6024
    ScaleWidth    = 11220
    StartUpPosition = 2 'CenterScreen
    Begin VB.Timer Timer1
        Interval    = 1000
        Left        = 7920
        Top         = 5280
    End
    Begin VB.Frame GatherData
        Caption     = "Gather Data"
        Height      = 972
        Left        = 240
        TabIndex    = 29
        Top         = 3600
        Width       = 3255
        Begin VB.CommandButton Pause
            Caption   = "Pause"
            Height    = 372
            Left      = 1800
            TabIndex  = 31
            Top       = 360
            Width     = 1092
        End
        Begin VB.CommandButton Begin
            Caption   = "Begin"
            Height    = 372
            Left      = 360
            TabIndex  = 30
            Top       = 360
            Width     = 1092
        End
    End
    Begin VB.Frame DataFile
        Caption     = "Data File"
        Height      = 1212
        Left        = 3960
        TabIndex    = 24
        Top         = 3480
        Width       = 6252
        Begin VB.OptionButton Append
            Caption   = "Append"
            ForeColor = &H000000FF&
            Height    = 192
            Index     = 1
            Left      = 240
            TabIndex  = 28
            Top       = 840
            Width     = 972
        End
        Begin VB.OptionButton Append
            Caption   = "Over Write"
            ForeColor = &H000000FF&
            Height    = 192
            Index     = 0
            Left      = 240
            TabIndex  = 27
            Top       = 600
            Width     = 1212
        End
    End
    Begin VB.TextBox ActiveFile
        Height      = 288
        Left        = 240
        TabIndex    = 26
        Text        = "Default.txt"
        Top         = 240
        Width       = 5772
    End
End

```

Listing 2—(continued)

```

    Begin VB.CommandButton SelfFile
        Caption     = "Select File"
        Height      = 372
        Left        = 2280
        TabIndex    = 25
        Top         = 720
        Width       = 1572
    End
End
Begin VB.Frame ModName
    Caption       = "Module Name"
    Height        = 855
    Left          = 4080
    TabIndex     = 17
    Top          = 2400
    Width        = 6255
    Begin VB.CommandButton ChangeModuleName
        Caption     = "Change"
        Height      = 375
        Left        = 4200
        TabIndex    = 19
        Top         = 360
        Width       = 1335
    End
    Begin VB.TextBox Text1
        Height      = 285
        Left        = 480
        TabIndex    = 18
        Text        = "MAN0"
        Top         = 360
        Width       = 3015
    End
End
Begin VB.Frame Rate
    Caption       = "Rate (0.5 to 3600 Sec/Sample)"
    Height        = 855
    Left          = 240
    TabIndex     = 16
    Top          = 2520
    Width        = 3255
    Begin VB.CommandButton SampleRate
        Caption     = "Update"
        Height      = 255
        Left        = 2160
        TabIndex    = 21
        Top         = 360
        Width       = 855
    End
    Begin VB.TextBox Text2
        Height      = 285
        Left        = 240
        TabIndex    = 20
        Text        = "1"
        Top         = 360
        Width       = 1575
    End
End
Begin VB.Frame Channels
    Caption       = "Channels to Record"
    Height        = 2055
    Left          = 240
    TabIndex     = 9
    Top          = 240
    Width        = 3252
    Begin VB.CheckBox Ch
        Caption     = "Channel 1"
        Height      = 195
        Index       = 0
        Left        = 240
        TabIndex    = 15
        Top         = 720
        Width       = 1200
    End
    Begin VB.CheckBox Ch
        Caption     = "Channel 2"
        Height      = 195
        Index       = 1
        Left        = 240

```

ated event and sets or clears the appropriate variables pretty much as you would expect by simple observation. If the user wishes to select a file name (including its path) different than the default, the SelectFile button loads the PickFile form (see Listing 3). That form manages the GUI process of selecting a path and file. The result is returned to the TopForm routine just like a subroutine call. I copied the PickFile form from Microsoft VB examples.

The heart of the TopForm is the timer event. Once every timer tick (I haven't yet figured out the exact setting), it enters the data-gathering routine. Although I expected data to be gathered and saved in this module by sending a command out the serial port and waiting for a reply, I was wrong. You shouldn't wait for the reply. It just bogs the computer down, which is now Win95 friendly. My plan instead is to either gather the data using the next timer as an interrupt or have a separate timer for reading the serial data.

I'll start with a separate timer for reading data so that I can operate the control for sending and reading at different rates. This approach also lets me have flags indicating if the data was sent or read without a transmission error.

Answer MAN is a low-cost serial ASCII module that packs together eight high-current parallel I/O lines, a 4-channel 8-bit ADC, a 2-channel 12-bit ADC, and a 2-channel 12-bit DAC (along with a set of powerful firmware functions like keypad scanning, 4 x 20 LCD control, analog limit monitoring, data averaging, frequency and event counting, PWM output, and reading Dallas iButton serial numbers) in a 28-pin DIP package. The serial communications can be either RS-232 or RS-485 format. For more information, check out the datasheet at (www.micromint.com).

A DLL, using a structure to pass data to and from the modules, helps connect the module with the VB programming environment. The prototype for the coding and the structure is found in the AnswerManStruct module, shown in Listing 4.

I've logically separated the communications into two parts. The first part initializes the DLL, sets up the hardware on the PC, and makes DLL calls. The second part manages what's connected to the serial port. If I send a command that has a reply, then I should get a reply or a flag indicating an error. Once you select a COM port, the error reporting starts. If you have the DLL installed, the status indicator turns green and the DLL version appears below the indicator.

Since I am only connecting one module, I won't use the RS-485 network. However, you program both RS-232 and RS-485 the same, so when I refer to RS-232, think both.

The Answer MAN protocol reads ASCII characters. To test error checking on the RS-232 end, I display the characters sent and received, which limits how fast I can gather the data. But, what the heck—it's good justification for a new computer, don't you think?

Listing 2—(continued)

```

        TabIndex      = 14
        Top           = 1005
        Width        = 1200
    End
    Begin VB.CheckBox Ch
        Caption       = "Channel 3"
        Height        = 195
        Index         = 2
        Left          = 240
        TabIndex      = 13
        Top           = 1290
        Width         = 1200
    End
    Begin VB.CheckBox Ch
        Caption       = "Channel 4"
        Height        = 195
        Index         = 3
        Left          = 240
        TabIndex      = 12
        Top           = 1590
        Width         = 1200
    End
    Begin VB.CheckBox Ch
        Caption       = "Channel 5"
        Height        = 195
        Index         = 4
        Left          = 1800
        TabIndex      = 11
        Top           = 720
        Width         = 1200
    End
    Begin VB.CheckBox Ch
        Caption       = "Channel 6"
        Height        = 195
        Index         = 5
        Left          = 1800
        TabIndex      = 10
        Top           = 1005
        Width         = 1200
    End
    Begin VB.Label Label2
        AutoSize      = -1 'True
        Caption       = "--- 12 Bit ---"
        Height        = 195
        Left          = 1920
        TabIndex      = 23
        Top           = 480
        Width         = 765
    End
    Begin VB.Label Label1
        AutoSize      = -1 'True
        Caption       = "--- 8 Bit ---"
        BeginProperty Font
            Name       = "MS Sans Serif"
            Size      = 9.6
            CharSet   = 0
            Weight    = 400
            Underline = 0 'False
            Italic    = 0 'False
            Strikethrough = 0 'False
        EndProperty
        Height        = 240
        Index         = 0
        Left          = 360
        TabIndex      = 22
        Top           = 457
        Width         = 825
    End
    End
    Begin VB.Frame Communications
        Caption       = "Communications"
        Height        = 2055
        Left          = 3960
        TabIndex      = 1
        Top           = 240
        Width         = 7095
        Begin VB.ComboBox BaudRateSel
            Height     = 288

```

Listing 2—(continued)

```

ItemData      = "TopForm.frx":0000
Left          = 1080
List          = "TopForm.frx":0016
TabIndex     = 32
Text         = "Default 9600"
Top          = 360
Width        = 1332
End
Begin VB.OptionButton ComSel
Caption      = "Com2"
Height      = 252
Index       = 1
Left        = 120
TabIndex    = 5
Top         = 480
Width       = 732
End
Begin VB.OptionButton ComSel
Caption      = "Com1"
Height      = 252
Index       = 0
Left        = 120
TabIndex    = 4
Top         = 240
Width       = 732
End
Begin VB.TextBox DataSent
BeginProperty Font
Name        = "MS Sans Serif"
Size        = 9.6
Charset     = 0
Weight      = 400
Underline   = 0 'False
Italic      = 0 'False
Strikethrough = 0 'False
EndProperty
Height      = 300
Left        = 2880
TabIndex    = 3
Text        = "Data Sent"
Top         = 240
Width       = 3975
End
Begin VB.TextBox Reply
BeginProperty Font
Name        = "MS Sans Serif"
Size        = 9.6
Charset     = 0
Weight      = 400
Underline   = 0 'False
Italic      = 0 'False
Strikethrough = 0 'False
EndProperty
Height      = 1275
Left        = 2880
MultiLine   = -1 'True
TabIndex    = 2
Text        = "TopForm.frx":003F
Top         = 600
Width       = 3975
End
Begin VB.Label CommEstab
AutoSize    = -1 'True
Caption     = "Status"
Height      = 195
Left        = 480
TabIndex    = 8
Top         = 840
Width       = 450
End
Begin VB.Shape ComOK
FillColor   = &H00404040&
FillStyle   = 0 'Solid
Height      = 210
Left        = 120
Shape       = 3 'Circle
Top         = 840
Width       = 210

```

Seriously, though, you can eliminate posting the RS-232 data at higher rates, gaining yourself a significant speed increase.

The other problem I wrestled with was data rate and file size. If you're running tests on a battery charger, you could be reading the data once per hour and running for several weeks. But, if you're running tests on a faster process, you need to read more often but only store the same quantity of readings. With no other direction, I'm just going to pick some numbers and hope they are work for your purposes. I will comment the code well, so you can see the limitations.

CREATING A PROTOTYPE THAT REALLY BOOKS

These are three approaches to software design: top down, bottom up, and what I call "mañana," which essentially postpones decisions until "tomorrow" or, more aptly, when you definitely know what to do.

The top-down approach starts with the big picture and designs the next lower level of the system on each pass through the design process. A bottom-up approach starts with the lowest level and builds the next higher level on each pass through the design process. The "mañana" approach postpones design decisions with the assumption that when issues affecting design need to be resolved, you'll postpone the decisions and keep on designing. Eventually, the reasons for not making the decision resolve, making it easier to make the correct design decision.

In our quest to build a generic data logger, I plan on taking the requirements outlined in Part 1 last month and get the data logger up and running. As you recall, I'm using a modular design written in Visual Basic and downloaded into Answer MAN, a module built by Micromint. I've decided to dub our generic data logger "Data MAN."

GETTING FUNCTIONAL

Getting this generic data logger together has been an interesting journey. As I started last month's article, I said to myself, "This is going to be fun and easy." I got a clean sheet of paper, thinking I could just write a paragraph for the article explaining what I'm doing and then write the code. I thought I could get twice as much done in half the time.

Obviously, I've been reading too many advertisements lately. And, if it had worked, it would have been a good description of a top-down design. I would have started with the big picture and kept going to lower and lower levels, designing and implementing along the way. At each level, I'd get into more details.

However, for this project, the top-down approach completely failed. I couldn't make any progress. I couldn't write code. It was all wrong. I've read that large systems are almost always designed and implemented using a top-down technique. If you're doing an air-traffic control system, for instance, you would need to complete the design and then test and simulate that design before you

Listing 2—(continued)

```

End
Begin VB.Label DLLVER
    AutoSize      = -1 'True
    Caption       = "Label1"
    Height        = 195
    Left          = 120
    TabIndex      = 7
    Top           = 1200
    Width         = 480
End
Begin VB.Label Aman
    AutoSize      = -1 'True
    Caption       = "Answer Man Version ????"
    Height        = 195
    Left          = 120
    TabIndex      = 6
    Top           = 1560
    Width         = 1770
End
End
Begin VB.CommandButton Exit
    Caption       = "Exit"
    BeginProperty Font
        Name       = "MS Sans Serif"
        Size       = 12
        CharSet    = 0
        Weight     = 700
        Underline  = 0 'False
        Italic     = 0 'False
        Strikethrough = 0 'False
    EndProperty
    Height        = 735
    Left          = 9480
    TabIndex      = 0
    Top           = 5040
    Width         = 1455
End
End
Attribute VB_Name = "TopForm"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
'.....
' A Data Logger
'
'   Written by:   ESC Inc
'               Old Saybrook, CT 06475
'               860-395-1192
'
' Copyright (C) 1999 by ESC Inc. All Rights Reserved
' Author: George F. Martin Created 24 February 1999
'
' This form is used to set up the data gathering parameters
'.....
' CAUTION This module is NOT the final version.
'.....
' Public Variables
'
Public CH1 As Boolean ' Is channel enabled - 8 Bit Jr
Public CH2 As Boolean ' Is channel enabled - 8 Bit Jr
Public CH3 As Boolean ' Is channel enabled - 8 Bit Jr
Public CH4 As Boolean ' Is channel enabled - 8 Bit Jr
Public CH5 As Boolean ' Is channel enabled - 12 Bit Sr
Public CH6 As Boolean ' Is channel enabled - 12 Bit Sr

Public ModuleName As String
    ' Name of the Module we're talking to
Public AppendFlag As Boolean ' Append Flag

```

Listing 2—(continued)

```

Option Explicit

Const COM_NONE = 0
Const COM1 = 1
Const COM2 = 2

Dim CmdOut As String ' The output string
Dim Line1 As String
Dim Line2 As String
Dim Line3 As String
Dim Line4 As String

Dim NewBaud As Integer ' The Baud Rate
Dim SampleRateSec As Double ' The sampling Rate
Dim ComPort As Integer ' Which comport to use
Dim GotPortOpen As Integer ' Flag we've got on opened
Dim Result As Integer ' Place holder for DLL calls
Dim TimeCount As Integer
    ' incremented every Timer1 interrupt
Dim DataGatherFlag As Boolean ' data gathering flag
Dim WaitCount As Integer
'.....
' The user changed the OverWrite-Append Selection
Private Sub Append_Click(Index As Integer)
    Select Case (Index)
        Case 0
            Append(0).ForeColor = COLOR_BLACK
            Append(1).ForeColor = COLOR_BLACK
            AppendFlag = False
        Case 1
            Append(0).ForeColor = COLOR_BLACK
            Append(1).ForeColor = COLOR_BLACK
            AppendFlag = True
        Case Else ' Other values.
            Debug.Print "Not a valid Append Index"
    End Select
End Sub

'.....
' The user has changed the Baud Rate
Private Sub BaudRateSel_Click()
    Dim temp As Integer

    temp = BaudRateSel.ListIndex ' Get the selected index.
    ' Index BaudRate
    ' 0 300
    ' 1 1200
    ' 2 2400
    ' 3 9600
    ' 4 19200
    ' 5 57600

    Select Case (temp)
        Case 0
            NewBaud = 300
        Case 1
            NewBaud = 1200
        Case 2
            NewBaud = 2400
        Case 3
            NewBaud = 9600
        Case 4
            NewBaud = 19200
        Case 5
            NewBaud = 57600

        Case Else ' Other values.
            Debug.Print "Not a Valid Baud Rate"
    End Select

```

Listing 2—(continued)

```

If (GotPortOpen = 1) Then
    Call amPortClose      ' Close the opened port
End If
' Try to open the port with the GreenLeaf DLL
Result = amPortOpen(ComPort, NewBaud)

If Result <> 0 Then
    MsgBox "Can Not Open Comm Port"
    ComPort = COM_NONE
Else
    ' Port Opened so get a pointer
    ' to a Structure in the DLL
    amData = amInitDLL(amData)
    ' all this is ByRef
    amData.ReadDelay = 30
    ' Set delay to a small value for this program

    If (amData.Status <> 0) Then
        ' This should never fail!!
        MsgBox "Can Not Initialize Data Structure"
        ComPort = COM_NONE
    Else
        GotPortOpen = 1
    End If
End If

End Sub

' Set the Data Gathering Flag
Private Sub Begin_Click()
    DataGatherFlag = True ' Set the data gathering flag
End Sub

' User clicked the Channel Enable Disable Check Box
Private Sub Ch_Click(Index As Integer)
    ' This isn't clever just simple
    Select Case (Index + 1) ' Evaluate Index
        Case 1 ' Toggle CHannel 1
            If (CH1 = True) Then
                CH1 = False
            Else
                CH1 = True
            End If
        Case 2 ' Toggle CHannel 2
            If (CH2 = True) Then
                CH2 = False
            Else
                CH2 = True
            End If
        Case 3 ' Toggle CHannel 3
            If (CH3 = True) Then
                CH3 = False
            Else
                CH3 = True
            End If
        Case 4 ' Toggle CHannel 4
            If (CH4 = True) Then
                CH4 = False
            Else
                CH4 = True
            End If
        Case 5 ' Toggle CHannel 5
            If (CH5 = True) Then
                CH5 = False
            Else
                CH5 = True
            End If
        Case 6 ' Toggle CHannel 6
    
```

ever started implementation. But, in my case, I was forcing myself into certain implementation details that I couldn't see or imagine until they were right on top of me.

So, then I switched to a bottom-up approach for the implementation of the data logger. I kept the design that the top-down approach created. That phase of the design process was completed rapidly and didn't force me into any dead ends. Implementing from the bottom up showed how fast, efficient, and flexible Visual Basic code can be. I could implement changes at low levels and not affect the overall design. Also, I could implement one general approach that met all the requirements and keep a clear path for a user to modify my work.

VISUAL BASIC'S ROLE

Some of the reasons these changes are possible hinge on how Visual Basics works. A VB program is a collection of routines, which execute based on events (e.g., timer ticks, mouse actions, data changes, button presses, and many, many other types). Most, if not all, of these events are operating-system generated. The VB program is entered, variables set up, screens displayed, and control returns to the operating system, and your program is ready to process the next event. This is a drastic change in how I used to write embedded software. It felt strange.

The heart of the implementation is the Timer1 routine (feel free to [download the source code](#) and play). This routine is entered every timer tick. The timer ticks change on the top screen to match the desired data-logging rate. Communications with Answer MAN take place through the serial port. If a communications port is opened, then we first try to read the data. The Answer MAN DLL accumulates serial data received in a buffer.

The call to the DLL, `ex_read_line()`, extracts a complete line of data. If a line is ready, it is read and displayed in a scrolling display. That's what the Line1, Line2, Line3, and Line4 variables are used for. Although this string manipulation offers great debugging assistance, it slows the program down considerably. My development system is a 200-MHz AMD K2 running Windows 95 with 64-MB RAM. I could run the program (V0.03) at two samples per second, and it would keep up.

Once a line is read and displayed and the DataGatherFlag is set, the line is then processed to extract any data returned from Answer MAN. Again, as you can see in [Listing 5](#) and [Figure 1](#), I read all six analog channels in this design. If the channel has been selected for logging from the top form, then I extract the reading. Since the A/D converters on Answer MAN operate from 0 to 255 or from 0 to 4095, I set all channel readings to -1 before I process the returned values. I gather all the data and write the data to a file.

The file format is also simple. Comments are identified with a semicolon at the beginning of the line. The

Listing 2—(continued)

```

        If (CH6 = True) Then
            CH6 = False
        Else
            CH6 = True
        End If

        Case Else ' Other values.
            Debug.Print "Not a Valid Channel to record"
        End Select
    End Sub

    .....
    ' The user requests a new module name
    Private Sub ChangeModuleName_Click()

        ModuleName = Text1.Text
        Text1.ForeColor = COLOR_BLACK
        ChangeModuleName.Visible = False

    End Sub

    .....
    ' Select a comm port 1 or 2 open it and try to use it.
    Private Sub ComSel_Click(Index As Integer)

        If (Index = 0) Then
            If ComPort = COM1 Then
                End ' Do nothing
            End If
            ComPort = COM1 ' Select com1:

        ElseIf (Index = 1) Then
            If ComPort = COM2 Then
                End ' Do Nothing
            End If
            ComPort = COM2 ' Select com2:
        End If

        If ComPort <> COM_NONE Then ' We have a port to try
            If (GotPortOpen = 1) Then
                Call amPortClose ' Close the opened port
            End If
            ' Try to open the port with the GreenLeaf DLL
            Result = amPortOpen(ComPort, NewBaud)

            If Result <> 0 Then
                MsgBox "Can Not Open Comm Port"
                ComPort = COM_NONE
            Else
                ' Port Opened so get a pointer
                ' to a Structure in the DLL
                amData = amInitDLL(amData)
                ' all this is ByRef
                amData.ReadDelay = 30
                ' Set delay to a small value for this program

                If (amData.Status <> 0) Then
                    ' This should never fail!!
                    MsgBox "Can Not Initialize Data Structure"
                    ComPort = COM_NONE
                Else
                    GotPortOpen = 1
                End If
            End If
        End If

        If GotPortOpen = 1 Then ' If port is opened then enable stuff
            Send.Enabled = True

        Else
            ' Else disable all the stuff
            Send.Enabled = False
        End If

    End Sub

    .....
    ' End this form

```

Listing 2—(continued)

```

    Private Sub Exit_Click()
        End
    End Sub

    .....
    ' The startup code whenever this form loads
    Private Sub Form_Load()
        PickFile.DataFileName = "Default.txt"

        CH1 = False ' 8 bit channels
        CH2 = False
        CH3 = False
        CH4 = False

        CH5 = False
        CH6 = False

        NewBaud = 9600 ' Start here
        GotPortOpen = 0 ' No ports opened

        ' Set default Sampling rate

        SampleRateSec = 1# ' Start with 1 sample Per Sec
        Text2.Text = SampleRateSec
        Text2.ForeColor = COLOR_BLACK
        SampleRate.Visible = False

        ' Set default Module name

        ModuleName = "MAN0"
        Text1.Text = ModuleName
        Text1.ForeColor = COLOR_BLACK
        ChangeModuleName.Visible = False

        ' Set OverWrite-Append

        Append(0).ForeColor = COLOR_RED
        Append(1).ForeColor = COLOR_RED

        CmdOut = "QA"

    End Sub

    .....
    ' Reset the data Gathering Flag
    Private Sub Pause_Click()

        DataGatherFlag = False ' Set the data gathering flag

    End Sub

    .....
    ' The user has requested a new rate
    Private Sub SampleRate_Click()
        Dim temp As Integer

        SampleRateSec = Val(Text2.Text)
        If (SampleRateSec < 0.5) Then ' Too fast
            temp = MsgBox("Sampling Rate Too Fast (0.5 to 3600)")
            SampleRateSec = 1#
        ElseIf (SampleRateSec > 3600) Then ' Too Slow
            temp = MsgBox("Sampling Rate Too Slow (0.5 to 3600)")
            SampleRateSec = 1#
        End If

        Text2.Text = SampleRateSec
        Text2.ForeColor = COLOR_BLACK
        SampleRate.Visible = False
        Timer1.Interval = SampleRateSec * 1000

    End Sub

    .....
    ' Pick a file to save the data into
    Private Sub SelFile_Click()
        PickFile.Show ' Showing it does the trick

    End Sub

```

Listing 2—(continued)

```

.....
' The user is changing the module name
Private Sub Text1_Change()
If (Text1.Text) = ModuleName Then
    Text1.ForeColor = COLOR_BLACK
    ChangeModuleName.Visible = False
Else
    Text1.ForeColor = COLOR_RED
    ChangeModuleName.Visible = True
End If
End Sub
.....
' The user changed the sample Rate so Prompt to update
Private Sub Text2_Change()
If (Val(Text2.Text)) = SampleRateSec Then
    Text2.ForeColor = COLOR_BLACK
    SampleRate.Visible = False
Else
    Text2.ForeColor = COLOR_RED
    SampleRate.Visible = True
End If
End Sub
.....
' Once every Timer1 interrupt check on
configuration & communication or get data
Private Sub Timer1_Timer()
Dim i As Integer
Dim temp As Integer

TimeCount = TimeCount + 1 ' Inc this counter
If TimeCount > 100 Then ' Just busy work
    TimeCount = 0 ' Keeps the variable from overflowing
End If

ComOK.FillColor = COLOR_GRAY

' Is this port opened
If GotPortOpen = 1 Then ' yes
    PrintButton.Enabled = True
    ' Enable the print/save button
Else
    PrintButton.Enabled = False
    ' Disable the print/save button
End If

' Is this port opened
If GotPortOpen = 1 Then ' yes
    If WaitCount > 0 Then ' Are we to wait
        WaitCount = WaitCount - 1 ' Yes
    Else ' Look to send any data
        If Len(CmdOut) > 0 Then ' Message present
            amData.TextOut = CmdOut & vbCrLf
            ' Get the Command
            DataSent.Text = CmdOut
            temp = ex_xmit_msg(amData)
            WaitCount = 1
        End If
    End If

    temp = 1
    While (temp > 0)
        ' Any Received Data
        temp = ex_read_line(amData)
        ' This has a built in pause
        If temp > 0 Then
            Line1 = Line2
            Line2 = Line3
            Line3 = Line4
        End If
    End While
End Sub

```

only other record type is a data record. All six fields are written out. If a field is not selected, the -1 is saved as a data value. On opening the file, I log some comments about the program and the current settings. And, on ending the data gathering, I write some closing remarks before closing the file.

After the data is read and processed in the Timer1 routine, the command to get the next set of readings is sent to Answer MAN via `ex_xmit_msg()`.

And finally, I monitor the status flags generated in the DLL. If a problem is reported, I set an indicator to red. If there are no problems, I set it to green.

The top screen looks the same as in last month's article. Let's comment about some of the issues covered in the design of this form. I've provided a means to select a COMM channel (1 or 2), change the data rate, and alter the name of the Answer MAN module you're communicating with. All of this is fairly specific to the Answer MAN product. You will need to modify these for your specific implementation.

The channels to read are selected with radio buttons. (BTW, I *did not* prevent you from changing these settings while the data gathering is in progress.) If you want to have a more user-friendly design, this is an area to look at. I've also implemented a sampling rate of from 0.5 to 3600 s per sample. The `SampleRate_Click()` routine sets those specific limits, and the file for saving the data can be opened in the append or overwrite mode and can be located in any directory available.

PICKING UP THE PACE

As is, this setup should be really good at monitoring rather slow processes—like once per second or slower. It would work well for battery charging/discharging tests or for monitoring building temperature and humidity.

What if we wanted to go faster? Let me give you some areas to work on. First, one limit is the serial port with the data going back and forth. In this application, you see the following characters on the screen.

MAN0>

QA

A1:00A2:00A3:00A4:00A5:000A6:000

This exchange involves a total of 39 or more character to complete a cycle. At 9600 bps or about 1 ms per character, it would take about 40–50 ms just for the data. As well, Answer MAN has a delay built into the reply that can take up to approximately 255 ms. Answer MAN also needs to read and compose the response using another 10 ms.

So, conservatively, we have 310 ms (50 + 250 + 10) built in as a time limit. We can reduce the turn around in the Answer MAN from 250 to 100 ms. And, we can up the data rate. I would expect that we could drop these limits to under 200 ms, which would result in five readings per second with little effort.

Listing 2—(continued)

```

Line4 = amData.Reply

Reply.Text = Line4
' this works
'
Reply.Text = "123" & vbCrLf & "asd"

' Post the received data
' Parse the returned data
' look for configuration
If (Reply.Text Like "C:*") = True Then
Elseif ((Left(Reply.Text, 10) Like "Answer
MAN") = True) Then Aman.Caption =
Reply.Text
UpdateAMVersion = False
' We need to prompt for an update
End If

End If
Wend

If ((GotPortOpen = 1) And (amData.Status = 0)) Then
ComOK.FillColor = COLOR_GREEN ' Got Config Data
DLLVER.Caption = amData.ID
CommEstab.Caption = "Status OK"
Else
ComOK.FillColor = COLOR_RED 'Error with Config Data
DLLVER.Caption = "DLL Version Unknown"
CommEstab.Caption = "Status Errors"
End If
End If

' Check for Printing
'If PrintFlag = 1 Then
' DoPrint
' PrintFlag = 0
'End If

End Sub
..... End of File.....
RSION 5.00
Begin VB.Form PickFile
Caption = "Select a file"
ClientHeight = 4740
ClientLeft = 3312
ClientTop = 2616
ClientWidth = 6660
LinkTopic = "Form1"
PaletteMode = 1 'UseZOrder
ScaleHeight = 4740
ScaleWidth = 6660
Begin VB.CommandButton Command1
Caption = "Done"
Height = 372
Left = 4200
TabIndex = 7
Top = 3960
Width = 1572
End
Begin VB.TextBox Text1
Height = 288
Left = 3480
TabIndex = 3
Text = "Text1"
Top = 720
Width = 2775
End
Begin VB.FileListBox File1
Height = 2568
Left = 3480
TabIndex = 2
Top = 1080
Width = 2772
End
Begin VB.DirListBox Dir1
Height = 1830
Left = 480
TabIndex = 1
Top = 1560
Width = 2415

```

Listing 2—(continued)

```

End
Begin VB.DriveListBox Drive1
Height = 315
Left = 480
TabIndex = 0
Top = 720
Width = 2415
End
Begin VB.Label Label13
AutoSize = -1 'True
Caption = "Files"
BeginProperty Font
Name = "MS Sans Serif"
Size = 9.6
Charset = 0
Weight = 400
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Height = 240
Left = 3480
TabIndex = 6
Top = 480
Width = 432
End
Begin VB.Label Label12
AutoSize = -1 'True
Caption = "Directories"
BeginProperty Font
Name = "MS Sans Serif"
Size = 9.6
Charset = 0
Weight = 400
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Height = 240
Left = 480
TabIndex = 5
Top = 1320
Width = 996
End
Begin VB.Label Label11
AutoSize = -1 'True
Caption = "Drives"
BeginProperty Font
Name = "MS Sans Serif"
Size = 9.6
Charset = 0
Weight = 400
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Height = 240
Left = 480
TabIndex = 4
Top = 480
Width = 588
End
End
Attribute VB_Name = "PickFile"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
'
' A Data Logger
'Written by:ESC Inc.,Old Saybrook, CT 06475, 860-395-1192
' Copyright (C) 1999 by ESC Inc. All Rights Reserved
' Author: George F. Martin, Created 24 February 1999
' This form is used to pick a file to save the data
into
'
.....
' CAUTION This module is NOT the final version.

```

Listing 2—(continued)

```
' .....  
' This file name is ont one selected  
Public DataFileName As String ' Contained the name of the  
Data File  
  
Option Explicit  
  
' The DONE button  
Private Sub Command1_Click()  
DataFileName = File1.Path + "\" + Text1.Text ' Build the  
name  
    PickFile.Hide ' Just hide  
this form  
TopForm.ActiveFile.Text = PickFile.DataFileName  
End Sub  
  
' The Directories window  
Private Sub Dir1_Change()  
    File1.Path = Dir1.Path  
    Text1.Text = ""  
End Sub  
  
' The Drives window  
Private Sub Drive1_Change()  
    Dir1.Path = Drive1.Drive  
End Sub  
  
' The Files window  
Private Sub File1_Click()  
    Text1.Text = File1.filename  
End Sub  
  
' Select a file  
Private Sub File1_Db1Click()  
    DataFileName = File1.Path + "\" + Text1.Text  
    PickFile.Hide  
End Sub  
  
' Load this form  
Private Sub Form_Load()  
    Dir1.Path = Drive1.Drive  
    Text1.Text = ""  
End Sub
```

Can your PC keep up? If not, we can help out in this area also. First, let's streamline the data file by only processing and writing to the file the channels we're interested in. Secondly, the command to write to the file is done before we send the command for more data. If we transpose those operations, we make better use of time.

Another REALLY BIG help: reduce the amount of data recorded. When we read the data, let's only record the data that changes significantly. You can set these limits to suit your application.

Also, we can convert the units of the raw data. By doing this, the data in the file is converted to whatever unit is needed for final calculations (e.g., degrees or PSI).

I've read the data gathered with this code into an Excel spreadsheet. With Excel's charting wizard, I've created a chart that wasn't pretty, but it was fast. I could have spent more time to clean it up, but I'd rather get charting into this program, which is next month's topic, by the way.

So, feel free to download the code and poke about a bit. I've updated some of the files. Make sure you throw away the files you downloaded last month and start from scratch. And, I'll be back with fresher code and tips on charting next month.

SOFTWARE

Source code for this article will be revised as the other parts of this article are developed. [Download the](#)

Listing 3—*With the far-reaching touch of GUI interfaces, some forms have become universally known. PickFile is a good example of this. It's the form that manages the GUI process of selecting a path and file. You can copy it from the Microsoft VB examples..*

```
VERSION 5.00
Begin VB.Form PickFile
    Caption       = "Select a file"
    ClientHeight  = 4740
    ClientLeft    = 3312
    ClientTop     = 2616
    ClientWidth   = 6660
    LinkTopic     = "Form1"
    PaletteMode   = 1 'UseZOrder
    ScaleHeight   = 4740
    ScaleWidth    = 6660
    Begin VB.CommandButton Command1
        Caption     = "Done"
        Height      = 372
        Left        = 4200
        TabIndex    = 7
        Top         = 3960
        Width       = 1572
    End
    Begin VB.TextBox Text1
        Height      = 288
        Left        = 3480
        TabIndex    = 3
        Text        = "Text1"
        Top         = 720
        Width       = 2775
    End
    Begin VB.FileListBox File1
        Height      = 2568
        Left        = 3480
        TabIndex    = 2
        Top         = 1080
        Width       = 2772
    End
    Begin VB.DirListBox Dir1
        Height      = 1830
        Left        = 480
        TabIndex    = 1
        Top         = 1560
        Width       = 2415
    End
    Begin VB.DriveListBox Drive1
        Height      = 315
        Left        = 480
        TabIndex    = 0
        Top         = 720
        Width       = 2415
    End
    Begin VB.Label Label13
        AutoSize    = -1 'True
        Caption     = "Files"
        BeginProperty Font
            Name      = "MS Sans Serif"
            Size      = 9.6
            Charset   = 0
            Weight    = 400
            Underline = 0 'False
            Italic    = 0 'False
            Strikethrough = 0 'False
        EndProperty
        Height      = 240
        Left        = 3480
        TabIndex    = 6
        Top         = 480
        Width       = 432
    End
    Begin VB.Label Label2
        AutoSize    = -1 'True
        Caption     = "Directories"
        BeginProperty Font
            Name      = "MS Sans Serif"
            Size      = 9.6
            Charset   = 0
            Weight    = 400
            Underline = 0 'False
```

Listing 3—(continued)

```
        Italic    = 0 'False
        Strikethrough = 0 'False
    EndProperty
    Height      = 240
    Left        = 480
    TabIndex    = 5
    Top         = 1320
    Width       = 996
End
Begin VB.Label Label1
    AutoSize    = -1 'True
    Caption     = "Drives"
    BeginProperty Font
        Name      = "MS Sans Serif"
        Size      = 9.6
        Charset   = 0
        Weight    = 400
        Underline = 0 'False
        Italic    = 0 'False
        Strikethrough = 0 'False
    EndProperty
    Height      = 240
    Left        = 480
    TabIndex    = 4
    Top         = 480
    Width       = 588
End
Attribute VB_Name = "PickFile"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
' A Data Logger
'Written by: ESC Inc.,Old Saybrook,CT 06475, 860-395-1192
' Copyright (C) 1999 by ESC Inc. All Rights Reserved
' Author: George F. Martin, Created 24 February 1999
' This form is used to pick a file to save the data into
'
' CAUTION This module is NOT the final version.
' This file name is ont one selected
Public DataFileName As String 'Containd the name of the
Data File
Option Explicit

' The DONE button
Private Sub Command1_Click()
    DataFileName = File1.Path + "\" + Text1.Text ' Build the
name
    PickFile.Hide ' Just hide
this form
    TopForm.ActiveFile.Text = PickFile.DataFileName
End Sub
' The Directories window
Private Sub Dir1_Change()
    File1.Path = Dir1.Path
    Text1.Text = ""
End Sub
' The Drives window
Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive
End Sub
' The Files window
Private Sub File1_Click()
    Text1.Text = File1.filename
End Sub
' Select a file
Private Sub File1_DblClick()
    DataFileName = File1.Path + "\" + Text1.Text
    PickFile.Hide
End Sub
' Load this form
Private Sub Form_Load()
    Dir1.Path = Drive1.Drive
    Text1.Text = ""
End Sub
```

Listing 4—Every module uses a structure to pass data to and from the modules, which helps connect it to the VB programming environment. The prototype for the coding and the structure is found in the AnswerManStruct module.

```

Attribute VB_Name = "AnswerManStruct"
' Written by: ESC Inc., Old Saybrook, CT 06475, 860-395-1192
' Copyright (C) 1999 by ESC Inc. All Rights Reserved
' Author: George F. Martin    Created 25 February 1999

' DLL Procedures
'
Declare Function ex_xmit_msg Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct) As Integer
Declare Function ex_read_line Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct) As Integer

Declare Function amPortOpen Lib "C:\AnswerMan.dll" (ByVal WhichPort As Integer, ByVal Baud As Integer) As Integer
Declare Function amInitDLL Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct) As amDLLStruct
Declare Function amGetConfiguration Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct) As Integer
Declare Sub amPortClose Lib "C:\AnswerMan.dll" ()
Declare Function amGetAllIO Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct) As Integer
Declare Function amGetDigitalDirection Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct) As Integer
Declare Function amSetAnalogOut Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct, ByVal n0 As Integer, ByVal n1 As Integer) As Integer
Declare Function amSetAnalogLimits Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct, ByVal Ch As Integer, ByVal Lo As Integer, ByVal Hi As Integer) As Integer
Declare Function amGetMinMax Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct) As Integer
Declare Function amResetMinMax Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct) As Integer
Declare Function amSetPWM Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct, ByVal tp As Long, ByVal hp As Long) As Integer
Declare Function amGetFrequency Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct) As Integer
Declare Function amResetTotalizer Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct) As Integer
Declare Function amGetTotalizer Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct) As Integer
Declare Function amSetDebounce Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct, ByVal Prd As Integer) As Integer
Declare Function amSetDigitalOut Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct, ByVal DigOut As Integer) As Integer
Declare Function amSetDirection Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct, ByVal DigDir As Integer) As Integer
Declare Function amGetDigitalOut Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct) As Integer
Declare Function amGetDigitalIn Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct) As Integer
Declare Function amGetSerialNumber Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct) As Integer
Declare Function amPrint Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct) As Integer
Declare Function amKey Lib "C:\AnswerMan.dll" (ByRef amStruct As amDLLStruct) As Integer

Option Explicit

' The following is the structure defined in the DLL
Public Const IO_BUF_MAX = 100      ' Space for communications
Public amData As amDLLStruct
Public Type amDLLStruct

    ID As String * 40      ' AnswerMan DLL ID
    Name As String * 20    ' Network address
    Command As String * IO_BUF_MAX ' Command sent to the Answer Man
    Reply As String * IO_BUF_MAX  ' Reply received from the Answer Man
    Key As String * 20      ' Keyboard Buffer
    TextOut As String * IO_BUF_MAX ' Text to Send to the Printer or LCD

    Status As Integer      ' Reply Status
    Config As Long         ' Configuration Bit Map

    Bin0 As Integer        ' Analog Input 0 (8 bit)
    Bin1 As Integer        ' Analog Input 1 (8 bit)
    Bin2 As Integer        ' Analog Input 2 (8 bit)
    Bin3 As Integer        ' Analog Input 3 (8 bit)

    Ain0 As Integer        ' Analog Input 0 (12 bit)
    Ain1 As Integer        ' Analog Input 1 (12 bit)

    ' These Thresholds are ONLY sent NEVER reported
    AnHiTh(0 To 5) As Integer ' Hi Threshold Analog Ch 0-5
    AnLoTh(0 To 5) As Integer ' Lo Threshold Analog Ch 0-5

    Aout0 As Integer       ' Analog Output 0 Setting (12 bit)
    Aout1 As Integer       ' Analog Output 1 Setting (12 bit)

    Freq As Long           ' Frequency Input (16 bit)
    DDir As Integer        ' I/O Port Direction Bit Map
    DIn As Integer         ' Digital Inputs
    DOut As Integer        ' Digital Outputs

    Avg0 As Integer        ' Average Analog Input 0 (12 bit)
    Max0 As Integer        ' Maximum Analog Input 0 (12 bit)

```

Listing 4—(continued)

```

Min0 As Integer      ' Minimum Analog Input 0 (12 bit)

Avg1 As Integer      ' Average Analog Input 1 (12 bit)
Max1 As Integer      ' Maximum Analog Input 1 (12 bit)
Min1 As Integer      ' Minimum Analog Input 1 (12 bit)

PrinterStat As Integer ' Printer Status

T(0 To 7) As Integer ' Touch Memory Serial Number T[0..7]

Counter As Long      ' Totalizer Value
PWMtp As Long ' PWM Total Period Counts
PWMhp As Long ' PWM High Period Counts
DebouncePeriod As Integer ' PWM debounce period
ReadDelay As Integer ' Read Delay Value in ms

End Type

```

Listing 5—Visual Basic follows nontraditional programming methods, but it's simple and quick.

```

Attribute VB_Name = "FileIO"
' Written by: ESC Inc., Old Saybrook, CT 06475 860-395-1192
' Copyright (C) 1999 by ESC Inc. All Rights Reserved
' Author: George F. Martin Created 15 April 1999

' File Procedures

Option Explicit

' File Open
'
Sub OpenFile()

If AppendFlag = True Then
    Open TopForm.ActiveFile.Text For Append As #1 ' Open file for output.
Else
    Open TopForm.ActiveFile.Text For Output As #1 ' Open file for output.
End If

Print #1, ";;;;;;;;;;"
Print #1, ";;Answer Man Data Logger Version 00.04"
Print #1, ";;Opened @ Date: "; Date; "Time: "; Time
Print #1, ";;SamplePeriod: "; SampleRateSec; " (sec)"
Print #1, ";;;;;;;;;;"
FileIsOpen = True

End Sub

' File Close
'
Sub CloseFile()

Print #1, ";;;;;;;;;;"
Print #1, ";;Closed @ Date: "; Date; "Time: "; Time
Print #1, ";;;;;;;;;;"
Close #1 ' Close file.
FileIsOpen = False

End Sub

' Write the latest data to the file
'
Sub WriteFile()

Print #1, CH1Data, CH2Data, CH3Data, CH4Data, CH5Data, CH6Data

End Sub

' END OF FILE

```

Circuit Cellar, The Magazine for Computer Applications.
Reprinted by permission. For subscription information,
call (860) 875-2199 or subscribe@circuitcellar.com